A Brief Introduction to Diffusion Models[1,2]

Ding Bai

January 5th, 2023

[1] Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics".
[2] Ho, Jain, and Abbeel, "Denoising Diffusion Probabilistic Models".

Outline

Background

- Markov Chain Conditional probability (e.g. $A \to B \to C$):

$$\begin{aligned} P(A,B,C) &= P(C|B)P(B|A)P(A) \\ P(B,C|A) &= P(C|B)P(B|A) \end{aligned} \tag{1}$$

- KL-Divergence for Gaussian distributions, where $p = \mathcal{N}(\mu_1, \sigma_1^2)$ and $q = \mathcal{N}(\mu_2, \sigma_2^2)$:

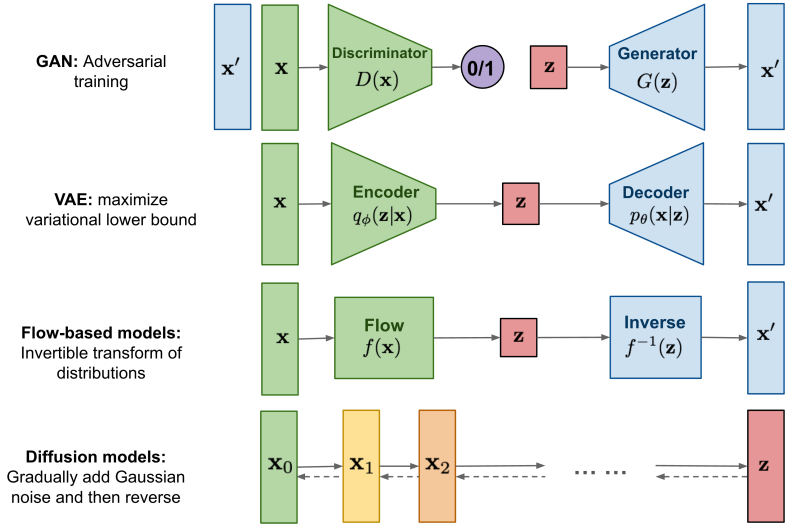
$$\begin{aligned} D_{KL}(p\|q) &= E_p[\log \frac{p}{q}] \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \end{aligned} \tag{2}$$

- Renormalization: to sample $x \sim \mathcal{N}(\mu, \sigma^2)$, where μ and σ are parameters, sample $z \sim \mathcal{N}(0,1)$ then $x := \mu + \sigma z$.

Diffusion probabilistic models

- A generative model. Unsupervised or Self-supervised.
- "Simultaneously achieves both flexibility and tractability"
- Tractable models: can be analytically evaluated (e.g. a Gaussian or Laplace); are not for rich datasets;
- Flexible models: can be molded to fit structure in arbitrary data; are not easy to analyze, and require Monte Carlo process.
- The first one (2015) is more abstract; Most subsequent works follow the DDPM (2020).

Relationship to other work: Generative models[3]



[3]Weng, "What are diffusion models?"

Relationship to other work: VAE

- Diffusion models: similar to VAE; let's recall it.
- Single-layer VAE: $\mathbf{x} \to \mathbf{z}$

$$
\begin{aligned}
p(\mathbf{x}) &= \int_{\mathbf{z}} p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) = \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \\
\log p(\mathbf{x}) &= \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\frac{p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}] \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log \frac{p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))
\end{aligned} \tag{3}
$$

- Multi-layer VAE: $\mathbf{x} \to \mathbf{z}_1 \to ... \to \mathbf{z}_n$.
  Replace $\mathbf{z}$ with $(\mathbf{z}_1, ..., \mathbf{z}_n)$ and use Markov Chain Conditions.

Relationship to other work: VAE

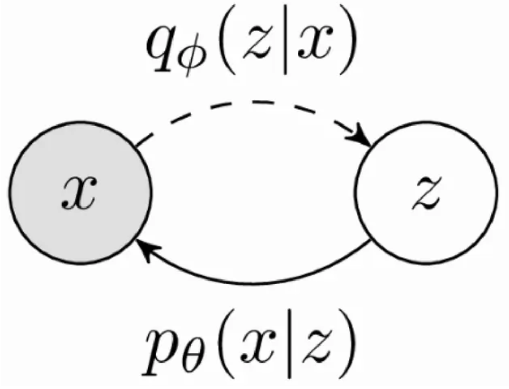


Figure: Single-layer VAE

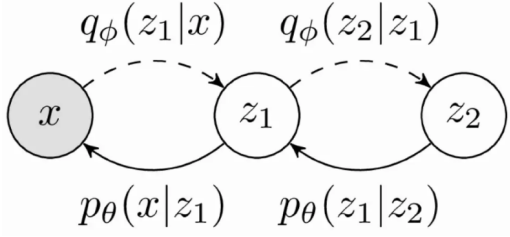
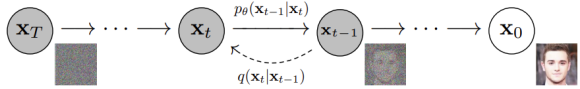
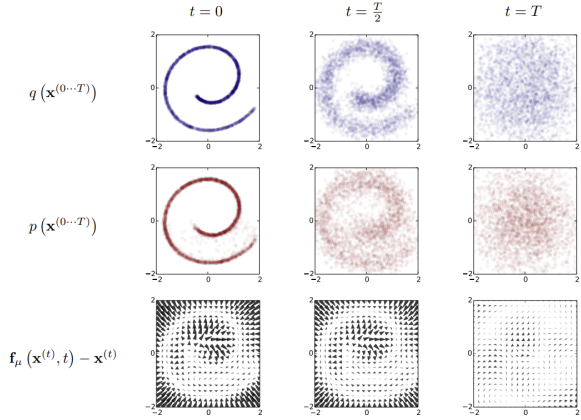


Figure: Multi-layer VAE

Overview





The bottom row shows the drift term for the same reverse diffusion process.

Forward Trajectory

- Basic form (in 2015 paper): repeated application of a Markov diffusion kernel $T_\pi(\mathbf{y}|\mathbf{y}';\beta)$.

$$\begin{aligned} \pi(\mathbf{y}) &= \int_{y'} T_\pi(\mathbf{y}|\mathbf{y}';\beta)\pi(\mathbf{y}') \\ q(\mathbf{x}_t|\mathbf{x}_{t-1}) &= T_\pi(\mathbf{x}_t|\mathbf{x}_{t-1};\beta_t) \\ q(\mathbf{x}_{1:T}|\mathbf{x}_0) &= \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \end{aligned} \tag{4}$$

- The kernel can be Gaussian or Binomial.

Forward Trajectory: in DDPM

- Gaussian and β_t settings (in 2020 paper DDPM):

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{5}$$

- The forward process variances β_t can be learned by reparameterization or held constant as hyperparameters. Usually: $\beta_1 < \beta_2 < ... < \beta_T$.
- It admits sampling $\mathbf{x}_t$ in closed form, where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && \text{;where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} && \text{;where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \\
q(\mathbf{x}_t|\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})
\end{aligned} \tag{6}$$

Reverse Trajectory

- The generative distribution will be trained to describe the same trajectory, but in reverse:

$$
\begin{aligned}
p(\mathbf{x}_T) &= \pi(\mathbf{x}_T) \\
p_\theta(\mathbf{x}_{0:T}) &= p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \\
p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \text{ or } \mathcal{B}(\mathbf{x}_{t-1}; b_\theta(\mathbf{x}_t, t))
\end{aligned}
\tag{7}
$$

- For both Gaussian and binomial diffusion, for continuous diffusion (limit of small step size β) the reversal of the diffusion process has an identical functional form as the forward process (Feller, 1949)[4].
- For all results in this (2015) paper, multi-layer perceptrons are used to define these functions (μ, Σ, b).

[4]Feller, "On the theory of stochastic processes, with particular reference to applications".

Reverse Trajectory: DDPM

- DDPM (2020) gives the closed form of the reverse conditional probability:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \tag{8}$$

$$\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\
&\propto \exp\Big(-\frac{1}{2}\big(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big) \\
&= \exp\Big(-\frac{1}{2}\big(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big) \\
&= \exp\Big(-\frac{1}{2}\big((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})\mathbf{x}_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)\mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\big)\Big)
\end{aligned}$$

- Remember: $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t)$. We have:

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t); \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{9}$$

the Negative Log-Likelihood

- Similar to VAE: add a KL-divergence

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{KL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}]
\end{aligned} \tag{10}
$$

Then we can use the Variational Lower Bound to optimize the negative log-likelihood:

$$
L_{VLB} = \mathbb{E}_{q(\mathbf{x}_{0:T})}[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}] \geq \mathbb{E}_{q(\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0)] = L_{CE} \tag{11}
$$

- If we use Jensen's inequality we can get the same result.

Lower bound on log-likelihood (2015)

- Result:

$$\begin{aligned} K = & -\sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_0,\mathbf{x}_t)}[D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))] \\ & + H_q(\mathbf{X}_T|\mathbf{X}_0) - H_q(\mathbf{X}_1|\mathbf{X}_0) - H_p(\mathbf{X}_T) \\ K <= & - L_{CE} \end{aligned} \tag{12}$$

- Training consists of finding the reverse Markov transitions which maximize this lower bound on the log-likelihood:

$$\hat{p}_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \arg \max_{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} K \tag{13}$$

Lower bound on log-likelihood (DDPM, 2020)

$$
\begin{aligned}
& L_{VLB} \\
& = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\
& = \mathbb{E}_{q}\Big[\log \frac{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
& = \mathbb{E}_{q}\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
& = \mathbb{E}_{q}\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
& = \mathbb{E}_{q}[\underbrace{D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T}\underbrace{D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}
\tag{14}
$$

Lower bound on log-likelihood (DDPM, 2020)

- Let's label each component in the variational lower bound loss separately:

$$
\begin{aligned}
L_{VLB} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\
L_t &= D_{KL}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1 \\
L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}
\tag{15}
$$

- Every KL term in L_{VLB} (except for L_0) compares two Gaussian distributions and therefore they can be computed in closed form.
- L_T is constant and can be ignored during training because q has no learnable parameters and x_T is a Gaussian noise. Ho et al. 2020 models L_0 using a separate discrete decoder derived from $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_\theta(\mathbf{x}_1, 1), \boldsymbol{\Sigma}_\theta(\mathbf{x}_1, 1))$.

Multiplying Distributions, and Computing Posteriors (2015)

- Tasks such as computing a posterior in order to do signal denoising or inference of missing values requires multiplication of the model distribution $p(\mathbf{x}_0)$ with a second distribution, or bounded positive function $r(\mathbf{x}_0)$:

$$\tilde{p}(\mathbf{x}_0) \propto p(\mathbf{x}_0) r(\mathbf{x}_0) \tag{16}$$

- Costly and difficult for many techniques (e.g. VAE, GSNs, NADEs, and most graphical models)
- However, under a diffusion model, it is straightforward since the second distribution can be treated either as a small perturbation to each step in the diffusion process, or often exactly multiplied into each diffusion step.

Multiplying Distributions, and Computing Posteriors: Results (2015)

The perturbed reverse diffusion kernel

- for Gaussian distributions:

$$\tilde{p}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\Big(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t) + \Sigma_\theta(\mathbf{x}_t, t)\frac{\partial \log r(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mu_\theta(\mathbf{x}_t,t)}, \Sigma_\theta(\mathbf{x}_t, t)\Big) \tag{17}$$

- for the perturbed Bernoulli: next slide.

Key equations in the paper (2015)

		Gaussian	*Binomial*
Well behaved (analytically tractable) distribution	$\pi\left(\mathbf{x}^{(T)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(T)}; \mathbf{0}, \mathbf{I}\right)$	$\mathcal{B}\left(\mathbf{x}^{(T)}; 0.5\right)$
Forward diffusion kernel	$q\left(\mathbf{x}^{(t)}\middle\|\mathbf{x}^{(t-1)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(t)}; \mathbf{x}^{(t-1)}\sqrt{1-\beta_t}, \mathbf{I}\beta_t\right)$	$\mathcal{B}\left(\mathbf{x}^{(t)}; \mathbf{x}^{(t-1)}\left(1-\beta_t\right) + 0.5\beta_t\right)$
Reverse diffusion kernel	$p\left(\mathbf{x}^{(t-1)}\middle\|\mathbf{x}^{(t)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(t-1)}; \mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right), \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right)$	$\mathcal{B}\left(\mathbf{x}^{(t-1)}; \mathbf{f}_b\left(\mathbf{x}^{(t)}, t\right)\right)$
Training targets		$\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right), \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right), \beta_{1\cdots T}$	$\mathbf{f}_b\left(\mathbf{x}^{(t)}, t\right)$
Forward distribution	$q\left(\mathbf{x}^{(0\cdots T)}\right) =$	$q\left(\mathbf{x}^{(0)}\right)\prod_{t=1}^{T} q\left(\mathbf{x}^{(t)}\middle\|\mathbf{x}^{(t-1)}\right)$	
Reverse distribution	$p\left(\mathbf{x}^{(0\cdots T)}\right) =$	$\pi\left(\mathbf{x}^{(T)}\right)\prod_{t=1}^{T} p\left(\mathbf{x}^{(t-1)}\middle\|\mathbf{x}^{(t)}\right)$	
Log likelihood	$L =$	$\int d\mathbf{x}^{(0)} q\left(\mathbf{x}^{(0)}\right)\log p\left(\mathbf{x}^{(0)}\right)$	
Lower bound on log likelihood	$K =$	$-\sum_{t=2}^{T}\mathbb{E}_{q\left(\mathbf{x}^{(0)},\mathbf{x}^{(t)}\right)}\left[D_{KL}\left(q\left(\mathbf{x}^{(t-1)}\middle\|\mathbf{x}^{(t)}, \mathbf{x}^{(0)}\right)\middle\|p\left(\mathbf{x}^{(t-1)}\middle\|\mathbf{x}^{(t)}\right)\right)\right] + H_q\left(\mathbf{X}^{(T)}\middle\|\mathbf{X}^{(0)}\right) - H_q\left(\mathbf{X}^{(1)}\middle\|\mathbf{X}^{(0)}\right) - H_p\left(\mathbf{X}^{(T)}\right)$	
Perturbed reverse diffusion kernel	$\tilde{p}\left(\mathbf{x}^{(t-1)}\middle\|\mathbf{x}^{(t)}\right) =$	$\mathcal{N}\left(x^{(t-1)}; \mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right) + \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\left.\frac{\partial \log r\left(\mathbf{x}^{(t-1)'}\right)}{\partial \mathbf{x}^{(t-1)'}}\right\|_{\mathbf{x}^{(t-1)'}=f_\mu\left(\mathbf{x}^{(t)}, t\right)}, \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right)$	$\mathcal{B}\left(x_i^{(t-1)}; \frac{c_i^{t-1}d_i^{t-1}}{x_i^{t-1}d_i^{t-1}+(1-c_i^{t-1})(1-d_i^{t-1})}\right)$

Table App.1. The key equations in this paper for the specific cases of Gaussian and binomial diffusion processes. $\mathcal{N}\left(u; \mu, \Sigma\right)$ is a Gaussian distribution with mean μ and covariance Σ. $\mathcal{B}\left(u; r\right)$ is the distribution for a single Bernoulli trial, with $u = 1$ occurring with probability r, and $u = 0$ occurring with probability $1 - r$. Finally, for the perturbed Bernoulli trials $b_i^t = \mathbf{x}^{(t-1)}\left(1-\beta_t\right) + 0.5\beta_t$, $c_i^t = \left[\mathbf{f}_b\left(\mathbf{x}^{(t+1)}, t\right)\right]_i$, and $d_i^t = r\left(x_i^{(t)} = 1\right)$, and the distribution is given for a single bit i.

Here it denotes μ_θ as f_μ, Σ_θ as f_Σ and b_θ as f_b for 3 parameterized functions.

Let's see the parameterization of DDPM

- Sohl-Dickstein et al. (2015) give perturbed equations for both Gaussian and Binomial distributions, but no specific algorithm.
- Ho et al. (DDPM, 2020) give a specific parameterization with detailed and simple enough training and sampling (generating) algorithms.
- Most following works are based on DDPM.

Forward Process and L_T

- Recall each component in the variational lower bound loss:

$$
\begin{aligned}
L_{VLB} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\
L_t &= D_{KL}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1 \\
L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}
\tag{18}
$$

- We ignore the fact that the forward process variances β_t are learnable by parameterization;
- And instead, fix them to constants (see next Section for details).
- Thus, in our implementation, the approximate posterior q has no learnable parameters
- so L_T is a constant during training and can be ignored.

Reverse Process and $L_{1:T-1}$

- Recall the KL divergence of 2 Gaussian p.d.f:

$$\begin{aligned} D_{KL}(p\|q) &= E_p[\log \frac{p}{q}] \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \end{aligned} \tag{19}$$

- We deparameterize $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$. Both choices of $\sigma_t^2 = \beta_t$ or $\tilde{\beta}_t$ had similar results.
- And then the parameterized L_t becomes (ignore the constant term while training):

$$\begin{aligned} L_{t-1} &= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\Big] \\ &= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\frac{1}{\sigma_t^2}\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big)\|^2\Big] \end{aligned} \tag{20}$$

Reverse Process: L_t

- The parameterized L_t (t from 1 to $T-1$) becomes:

$$\begin{aligned} L_{t-1} &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\sigma_t^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\right] \\ &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\sigma_t^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right] \end{aligned} \tag{21}$$

- Here we parameterize ϵ_θ as a function approximator intended to predict $\boldsymbol{\epsilon}_t$ from $\mathbf{x}_t$.
- We verify its effectiveness in the next section (experiments) in an ablation where they compare predicting $\boldsymbol{\epsilon}$ against predicting $\tilde{\boldsymbol{\mu}}_t$.

Data scaling, reverse process decoder, and L_0

- Assume that image data consists of integers in $\{0, 1, ..., 255\}$ scaled linearly to $[-1, 1]$, to fit into a the standard normal prior.
- To obtain discrete log-likelihoods, set the last term of the reverse process to an independent discrete decoder derived from the Gaussian $\mathcal{N}(\mathbf{x}_0; \mu_\theta(\mathbf{x}_1, 1), \sigma_1^2 \mathbf{I})$

$$p_\theta(\mathbf{x}_0|\mathbf{x}_1) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(\mathbf{x}_1, 1), \sigma_1^2)\, dx$$

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \qquad \delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}$$

- where D is the data dimensionality and the i superscript indicates extraction of one coordinate.
- It ensures that L_0 approximates as L_t so that we can obtain $\mu_\theta(\mathbf{x}_1, 1)$.
- There are better choices.

Simplified training objective

- Empirically, Ho et al. (2020) found it beneficial to sample quality to train on the following variant of the variational bound (ignoring the weight):

$$\begin{aligned} L_{\text{simple}}(\theta) &= \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\Big] \\ &= \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big] \end{aligned} \tag{22}$$

- where t is uniform between 1 and T.
- The $t = 1$ case corresponds to L_0 with the integral in the discrete decoder definition approximated by the Gaussian probability density function times the bin width, ignoring σ_1^2 and edge effects.

Final Algorithm

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

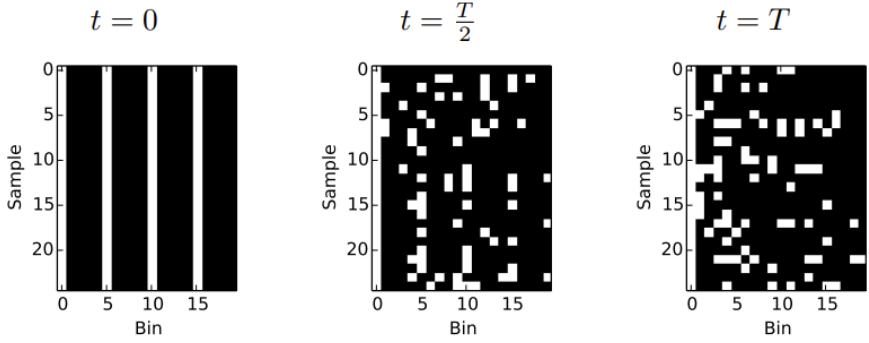
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Experiments (2015)

- Swiss roll distributions. See in previous page.
- Heartbeat distributions, Binomial.
  The log-likelihood under the true distribution: $\log_2 \frac{1}{5} = -2.322$ bits per sequence.
  Nearly perfect.

$p\left(\mathbf{x}^{(0\cdots T)}\right)$



- Also trained on several image datasets.

Experiments (2015) Tables

Here are tables of K by the diffusion model for different datasets, and comparisons of different models on the dataset of dead leaves images.

Dataset	K	$K - L_{null}$
Swiss Roll	2.35 bits	6.45 bits
Binary Heartbeat	-2.414 bits/seq.	12.024 bits/seq.
Bark	-0.55 bits/pixel	1.5 bits/pixel
Dead Leaves	1.489 bits/pixel	3.536 bits/pixel
CIFAR-10[3]	5.4 ± 0.2 bits/pixel	11.5 ± 0.2 bits/pixel
MNIST	See table 2	

Model	*Log Likelihood*
Dead Leaves	
MCGSM	1.244 bits/pixel
Diffusion	**1.489 bits/pixel**
MNIST	
Stacked CAE	174 ± 2.3 bits
DBN	199 ± 2.9 bits
Deep GSN	309 ± 1.6 bits
Diffusion	$\mathbf{317 \pm 2.7}$ **bits**
Adversarial net	325 ± 2.9 bits
Perfect model	349 ± 3.3 bits

Experiments (DDPM, 2020)

- hyper-parameters setting: $T = 1000$; β_t increases linearly from $\beta_0 = 10^{-4}$ to $\beta_T = 0.02$.
- Constants: small relative to data scaled to $[-1, 1]$, ensuring that reverse and forward processes have approximately the same functional form while keeping the signal-to-noise ratio at x_T as small as possible

$$L_T = D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|\mathcal{N}(0, \mathbf{I})) \approx 10^{-5}(\text{bits per dimension}) \tag{23}$$

- U-Net backbone similar to an unmasked PixelCNN++.
- Diffusion time t is specified by adding the Transformer sinusoidal position embedding into each residual block.
- Use self-attention at the 16×16 feature map resolution.

Experiments (DDPM, 2020) Tables

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

Table 2: Unconditional CIFAR10 reverse process parameterization and training objective ablation. Blank entries were unstable to train and generated poor samples with out-of-range scores.

Objective	IS	FID
$\tilde{\boldsymbol{\mu}}$ prediction (baseline)		
L, learned diagonal $\mathbf{\Sigma}$	7.28 ± 0.10	23.69
L, fixed isotropic $\mathbf{\Sigma}$	8.06 ± 0.09	13.22
$\|\tilde{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}_\theta\|^2$	–	–
$\boldsymbol{\epsilon}$ prediction (ours)		
L, learned diagonal $\mathbf{\Sigma}$	–	–
L, fixed isotropic $\mathbf{\Sigma}$	7.67 ± 0.13	13.51
$\|\tilde{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}_\theta\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	**3.17**

Conclusion

- Basic structure: Markov chain; Forward and reverse (VAE, but the forward process is not parameterized).
- Difference: the latent code has the same size as the input; (Latent diffusion model (2020) also proposed a compressed latent code[5])
- Algorithm intuition: put on random noises then recover (denoising); No actual chain is made.
- Paper in 2015 gives a basic structure; DDPM (2020) gives detailed algorithms.
- The high-quality generation of DDPM depends on a large T (≥ 1000), which causes the forward process of diffusion to be very slow. In the denoising diffusion implicit model (DDIM, 2020)[6], a means of reducing diversity in exchange for faster inference is proposed.

[5] Rombach et al., "High-resolution image synthesis with latent diffusion models".
[6] Song, Meng, and Ermon, "Denoising diffusion implicit models".

References

[1] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics.” ICML 2015.
[2] Jonathan Ho et al. “Denoising diffusion probabilistic models.” arxiv Preprint arxiv:2006.11239 (2020).
[3] Jiaming Song et al. “Denoising diffusion implicit models.” arxiv Preprint arxiv:2010.02502 (2020).
[4] Rombach, Robin, et al. ”High-Resolution Image Synthesis with Latent Diffusion Models.” arXiv preprint arXiv:2112.10752 (2021).
[5] Feller, William. ”On the theory of stochastic processes, with particular reference to applications.” Selected Papers I. Springer, Cham, 2015. 769-798.
[6] Weng, Lilian. (Jul 2021). What are diffusion models? Lil’Log. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.